On the case to expand PhySiMod with a EEO (Equation-based Object Oriented) Language

Vaibhav Arora, IFP School

Abstract

In the following work, the need of a library of reusable components and how acausal modeling naturally facilitates the same is discussed. Acausal modeling and signal-based approach for modeling are compared for physical system modeling. To that extent, limitations of signal-based approach are mentioned. An overview of working of EEO languages used in acausal modeling tools is discussed with suitable examples. Finally, with a view of the aforementioned, the compatibility of the PhySiMod environment with an existing EEO language tool (Simscape) is presented along with suitable reasons for extending the existing PhySiMod library with the same.

I. Introduction

For model development, in case of large-complex systems such as automotive that entail multiple domains, having a library of component models that describe the behavior of a small part of the larger system, is inherently desirable. For once a reusable set of behavioral model components has been created, the workflow allows the end-user of the models to work at a higher level of assembly and analysis of the system. Building systems with physical models involves simply connecting the needed components onto a schematic, all the while giving the user a physical perspective and not a mathematical one [1]. Making configuration changes becomes easy in such environment.

This is the essence of physical way of modeling, in which the model description innately encompasses modularity and reusability.

II. How physical modeling helps

Although there are many formalisms for modeling, considering white-box modeling so to speak, two of the most common approaches are

1. Causal or signal based models which are represented mathematically through explicit ordinary differential equations (ODEs) in state-space and transfer functions models on paper and through block diagrams graphically on computers.

2. Acausal or physical models, which are based on conservation principles and are mathematically represented through implicit differential-algebraic equations (DAEs) and bond-graphs graphically.

In causal modeling, using blocks, a system is graphically described in terms of ODEs and a software tool is then used for performing numerical integration. Example of tools include Simulink, SystemBuild. Any given block in a block diagram has the following general form:

$$\dot{x} = f(t, x, u) \qquad (1)$$

$$y = g(t, x, u) \qquad (2)$$

Where u represents the input signals, x represents the internal states and y represents the output signals.

In Simulink, a typical block stores the previous values of the state and/or inputs to compute the current values of the states. The Simulink 'integrator block' is an example of a 'block'. It outputs the integral of the input signal from the start of the simulation to the current time.

Block diagram approach assume that a system can be decomposed into block diagram structures with causal interactions. That is, one needs to know the particular input and output for which the model is being built. Although such an approach is natural for control system analysis, such causality assumptions at component levels limit the creation of libraries of basic elements, and thus hinders reusability of the component models at a system level. For example, consider a single resistor. Mathematically,

$$u(t) = R * i(t) \qquad (3)$$

but equally,

$$i(t) = \frac{1}{R} * u(t) \qquad (4)$$

Both models are obviously correct but the choice of the representation would depend on the instance of use (see Figure 1).

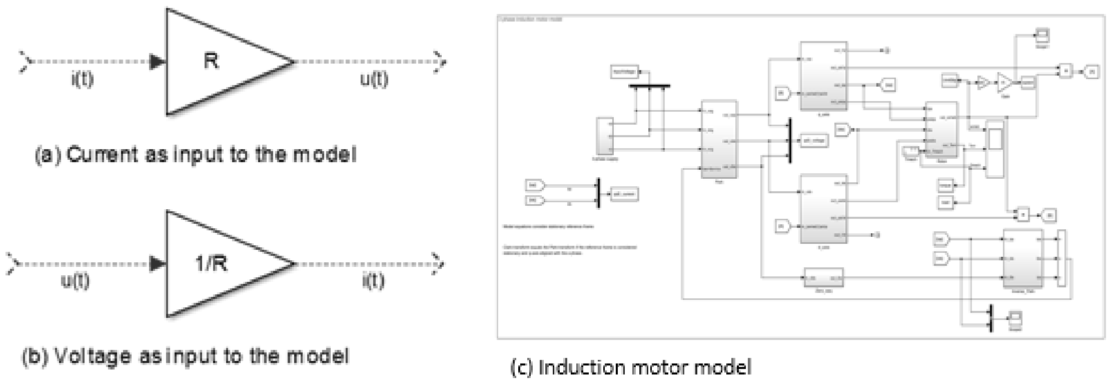

R
i(t)
u(t)
(a) Current as input to the model
1/R
u(t)
i(t)
(b) Voltage as input to the model
(c) Induction motor model

Fig. 1 Depiction of a causal approach

The aforementioned consideration is valid for other domains too. In the mechanical domain, consider a planetary gear-set whose behavior is represented by the following equations:

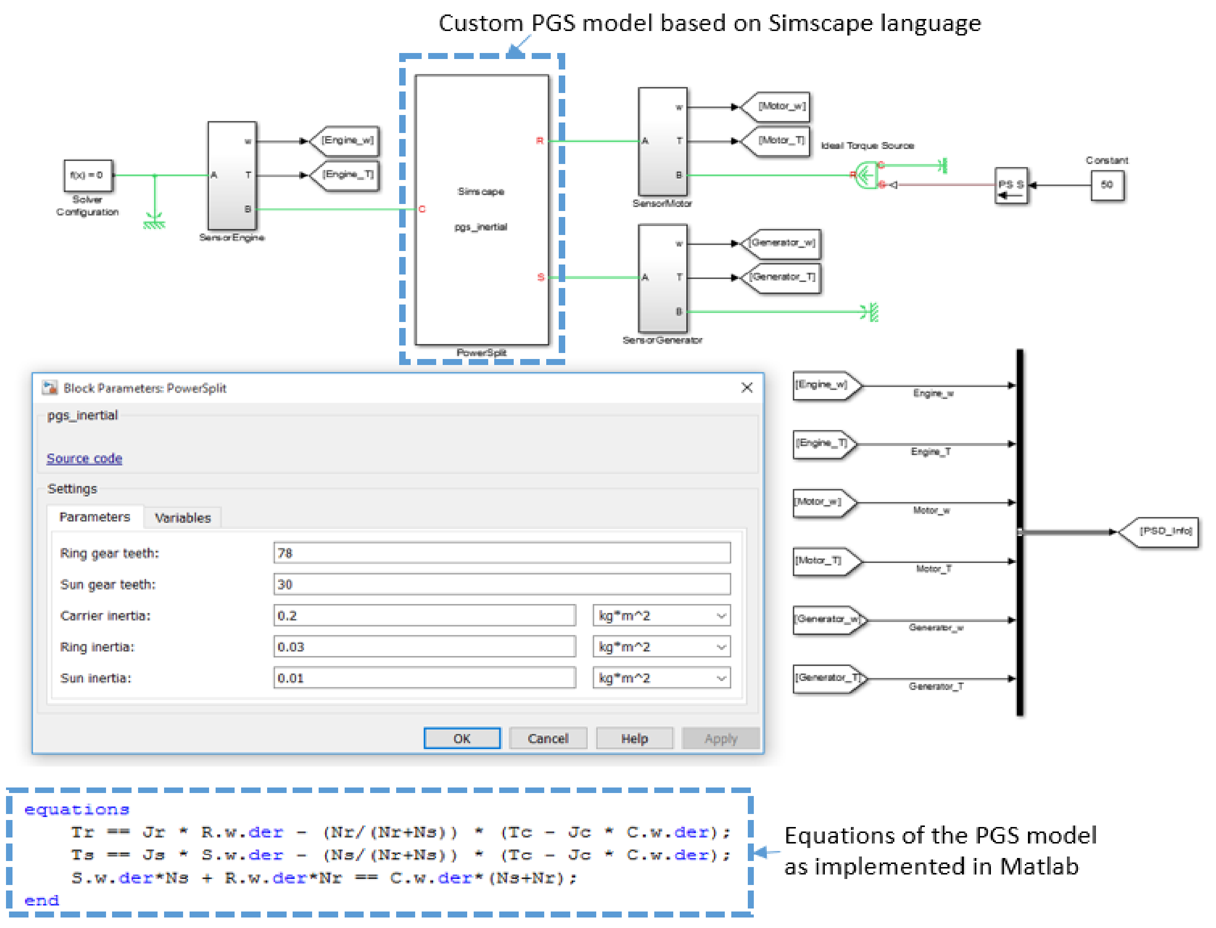
$$J_r \frac{d\omega_r}{dt} = T_r + \left(\frac{R}{R+S}\right)\left(T_c - J_c \frac{d\omega_c}{dt}\right) \qquad (5)$$

$$J_s \frac{d\omega_s}{dt} = T_s + \left(\frac{S}{R+S}\right)\left(T_c - J_c \frac{d\omega_c}{dt}\right) \qquad (6)$$

$$S\omega_s + R\omega_r = (S+R)\omega_c \qquad (7)$$

Depending on the arrangement, a planetary gear-set can have several different kinds of causality. Thus, state variables will differ with the arrangement in a typical model using the block-diagram based approach [2].

With an acausal modeling environment, a single component model can be used in a variety of contexts since no prior assumptions about the causality are required (see Figure 2).



Block Parameters: PowerSplit

pgs_inertial

Source code

Settings

Parameters | Variables

Ring gear teeth:	78	
Sun gear teeth:	30	
Carrier inertia:	0.2	kg*m^2
Ring inertia:	0.03	kg*m^2
Sun inertia:	0.01	kg*m^2

OK Cancel Help Apply

```
equations
    Tr == Jr * R.w.der - (Nr/(Nr+Ns)) * (Tc - Jc * C.w.der);
    Ts == Js * S.w.der - (Ns/(Nr+Ns)) * (Tc - Jc * C.w.der);
    S.w.der*Ns + R.w.der*Nr == C.w.der*(Ns+Nr);
end
```

Fig. 2 Simscape implementation of a planetary gear-box

Thus, for describing plant models or physical system behavior, block diagram formulations take more work to create (some mathematical manipulation is necessary to formulate the problem) and are less reusable [3].

In acausal formulation, there is no explicit specification of system inputs and outputs while building the model. Equivalently, the interactions among the models is realized not via signals but via energy exchange (see Figure 3). The link between the two subsystems must now be interpreted as a (bidirectional) exchange of energy between the two systems, flow of energy, power. Thus, having energy as the central object in modeling of elements allows them to be reusable at system level regardless of a priori description of the system, that is, to say, acausal modeling translates directly into a reusable set of library of physical components.

This also allows easy configuration changes at system levels. Another important feature of this approach is the inclusion of different physical domains with the same quantity.

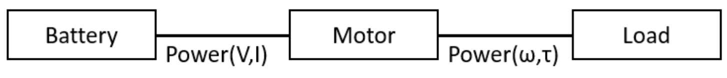


Fig. 3 The idea behind acausal modeling

III. How PhySiMod captures physical modeling in the block diagram environment of Simulink

The PhySiMod modeling convention defines boundary (holding flow information like Force/Torque) and state components (with state information like Linear/Angular Velocity) transferred exclusively through flow and state buses. Apart from this, Cumulative blocks and buses are included to account for the 'accumulating' mass/inertia. Simply putting, this accounts for the coupling effect of mass/inertia one would expect in physical systems. For example, the inertia of the wheel would impact the state of the driveline (see Figure 4).

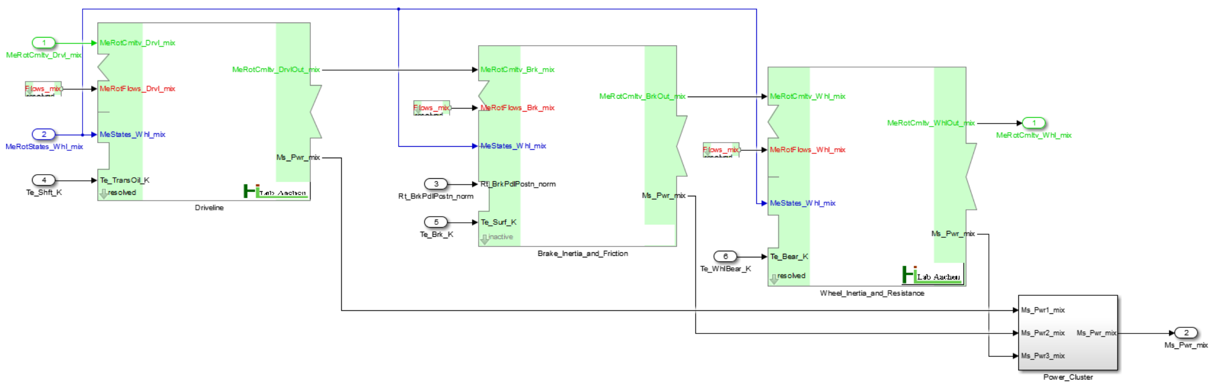


Fig. 4 A driveline model implementation in PhySiMod

The state blocks essentially integrate all flow and cumulative information in the buses to find the new state at each time-step.

Following such a convention, an efficient and real-time capable library of reusable physical components in mechanical (translational/rotational), thermal and thermal flow domain is available in the signal-based environment of Simulink.

IV. Limitations of signal based modeling tool solvers

Consider the circuit shown in Figure 5. The constitutive equations (like Ohm's law for a resistor) of the components are combined with conservation equations (like Kirchhoff's current law) to determine the overall system of equations.

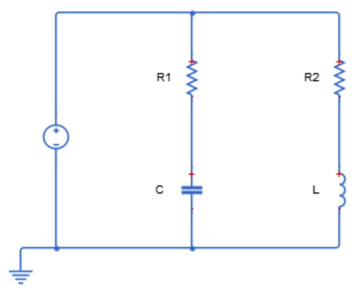


Fig. 5 A simple electrical circuit

Considering the model of the individual components,

$$u_0 = f(t) \qquad (8)$$

$$u_1(t) - R_1 i_1(t) = 0 \qquad (9)$$

$$u_2(t) - R_2 i_2(t) = 0 \qquad (10)$$

$$i_C(t) - C\frac{du_C(t)}{dt} = 0 \qquad (11)$$

$$u_L(t) - L\frac{di_L(t)}{dt} = 0 \qquad (12)$$

By application of Kirchoff's Voltage Law,

$$u_0 - u_1(t) - u_C(t) = 0 \qquad (13)$$

$$u_2(t) - u_L(t) + u_C(t) + u_1(t) = 0 \qquad (14)$$

By application of Kirchoff's Current Law,

$$i_0 - i_1(t) - i_2(t) = 0 \qquad (15)$$

$$i_1(t) = i_C(t) \qquad (16)$$

$$i_2(t) = i_L(t) \qquad (17)$$

Thus, an implicit DAE model is naturally obtained. The application of conservation laws, in general, result in systems of implicit DAEs of the form,

$$F\left(\frac{dx(t)}{dt}, x(t), u(t), t\right) = 0, \qquad x(t_0) = x_0 \qquad (18)$$

$$g(x(t), u(t), t) = 0 \qquad (19)$$

This is true not only for systems in electrical domain, but also in mechanical domain in rotational driveline systems (see Figure 6). Looking at the structure matrix of the electric circuit described in Figure 5, it can be seen that there is a necessity to solve a system of equations (see Eqn 20).

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad (20)$$

Such formulations result in algebraic-loops in signal based environment. However, equation based environment such as Simscape or Modelica are specialized at simulating systems described by differential-algebraic equations (DAE).

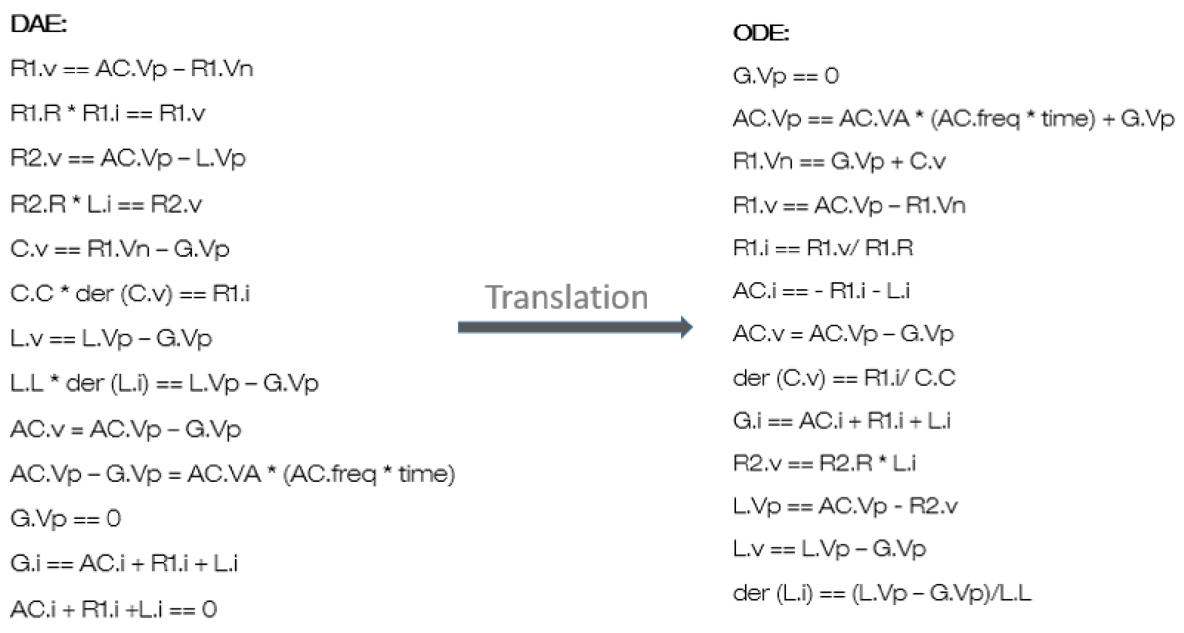
The translator in these modelling and simulation environment takes the described model as an input, generates a global system of DAEs and transforms the DAE to state space form by symbolic manipulation and graph theoretical algorithms. The original sorted equations contain a linear system of multiple simultaneous equations. Via tearing variables for algebraic loops (variables that reduce and decouple the equation system in such a way that there result separate equation systems in one tearing variable each) this system of equations is reduced to fewer linear equations which are solved by standard numerical procedures whenever the model is evaluated (see Section VI for more details).

The key is the principle of conservation of energy. To support this formalism, ports of a component defined are linked to a potential variable and a flow variable. If two or more ports are connected, the potential variables are set equal and the flow variables are summed to zero. Consider the node between the spring and the Rotor2 (see Figure 7). The following equations are implied:

$$\tau_{s2} + \tau_3 = 0 \qquad (21)$$

$$\phi_{s2} = \phi_2 \qquad (22)$$

Continuing with the example of the circuit in Figure 5, the translation process of acausal modeling tools would be as what follows. First the compiler would derive the equations from the graphical diagram of the model which would be in DAE form, and subsequently with the translation process, convert it into ODE from, if possible (see Figure 6)



```
DAE:
R1.v == AC.Vp – R1.Vn
R1.R * R1.i == R1.v
R2.v == AC.Vp – L.Vp
R2.R * L.i == R2.v
C.v == R1.Vn – G.Vp
C.C * der (C.v) == R1.i
L.v == L.Vp – G.Vp
L.L * der (L.i) == L.Vp – G.Vp
AC.v = AC.Vp – G.Vp
AC.Vp – G.Vp = AC.VA * (AC.freq * time)
G.Vp == 0
G.i == AC.i + R1.i + L.i
AC.i + R1.i +L.i == 0
```

Translation

```
ODE:
G.Vp == 0
AC.Vp == AC.VA * (AC.freq * time) + G.Vp
R1.Vn == G.Vp + C.v
R1.v == AC.Vp – R1.Vn
R1.i == R1.v/ R1.R
AC.i == - R1.i - L.i
AC.v = AC.Vp – G.Vp
der (C.v) == R1.i/ C.C
G.i == AC.i + R1.i + L.i
R2.v == R2.R * L.i
L.Vp == AC.Vp - R2.v
L.v == L.Vp – G.Vp
der (L.i) == (L.Vp – G.Vp)/L.L
```

Fig. 6 Equation manipulation during the 'Translation' process

V. Manual process

Before getting into the 'Translation' process, it is beneficial to think of the process one would go about manually in solving such equations, for the algorithms involved try to achieve the same efficiently and automatically. Once stating the general equations of the model, one would go about performing certain mathematical manipulation to get a state-space form to be able to use a numerical integration method. This might typically involve manipulation of individual equations to solve for the unknown, solving linear system of equations or non-linear with, differentiating certain equations (index reduction [8]).

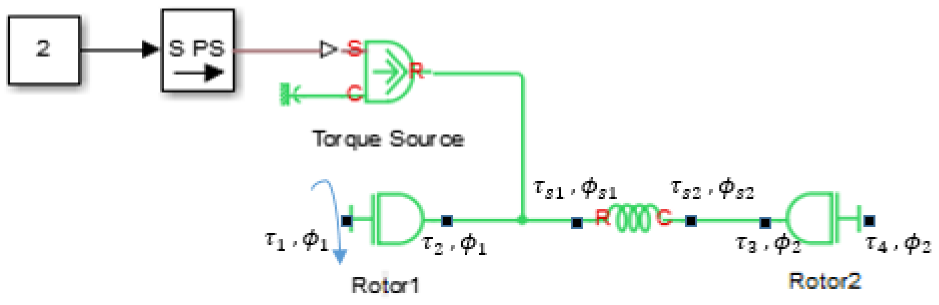


Fig. 7 A simple model representing a drive-shaft with a torque

As an example of the procedure, consider the simple system given in Figure 7 that results in a 0-order DAE system. Starting from the left-side, the model equations can be written as follows:

$$\phi_1 = \omega_1 \qquad (23)$$

$$\phi_2 = \omega_2 \qquad (24)$$

$$\dot{\omega}_1 = \frac{(\tau_1 + \tau_2)}{J_1} \qquad (25)$$

$$\dot{\omega}_2 = \frac{(\tau_3 + \tau_4)}{J_2} \qquad (26)$$

$$\tau_1 = u \quad (27)$$

$$\tau_2 = c(\phi_2 - \phi_1) \qquad (28)$$

$$\tau_3 = -\tau_2 \qquad (29)$$

$$\tau_4 = 0 \qquad (30)$$

The Structure Matrix for the system will be as given by Eqn 31.

$$\mathrm{P} = \begin{array}{c|cccccccc} & \dot{\phi}_1 & \dot{\phi}_2 & \dot{\omega}_1 & \dot{\omega}_2 & \tau_1 & \tau_2 & \tau_3 & \tau_4 \\ \hline \mathrm{f1} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathrm{f2} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathrm{f3} & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ \mathrm{f4} & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ \mathrm{f5} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \mathrm{f6} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \mathrm{f7} & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ \mathrm{f8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \qquad (31)$$

The order of the equations can be sorted to obtain a lower triangle matrix as given in Eqn 36. The system equations now have a causal form and solving them is straight forward. The algebraic terms can be eliminated and what would be left is a system of differential equations to be solved for some given values of c, J1, and J2.

$$\phi_1 = \omega_1 \qquad (32)$$

$$\phi_2 = \omega_2 \qquad (33)$$

$$\dot{\omega}_1 = \frac{(u + c(\phi_2 - \phi_1))}{J_1} \qquad (34)$$

$$\dot{\omega}_2 = \frac{(-c(\phi_2 - \phi_1))}{J_2} \qquad (35)$$

$$\mathrm{P}' = \begin{array}{c|cccccccc} & \tau_4 & \tau_3 & \tau_2 & \tau_1 & \dot{\omega}_2 & \dot{\omega}_1 & \dot{\phi}_2 & \dot{\phi}_1 \\ \hline \mathrm{f8} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathrm{f7} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathrm{f6} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \mathrm{f5} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \mathrm{f4} & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ \mathrm{f3} & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ \mathrm{f2} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \mathrm{f1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \qquad (36)$$

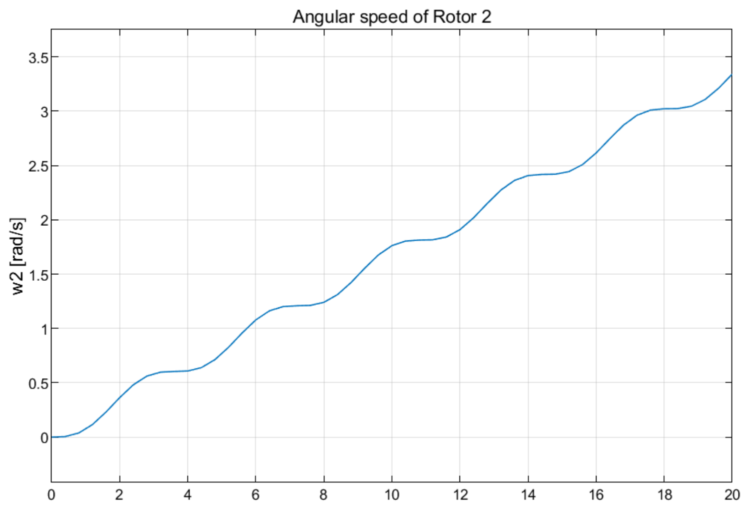
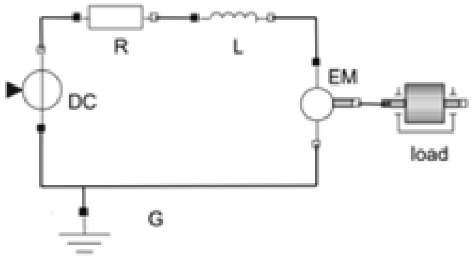


Fig. 8 The angular speed of rotor 2 [J1 = 10; J2 = 2; c = 5]

But sorting might not always result in a lower triangle matrix, for example, in the case of Index-1 DAEs. Such a system contains algebraic loops and have dedicated local solvers.

VI. Details of the Translator

As seen from the previous examples, it is clear that physical models are mapped into a mathematical description as a system of DAEs. This is done automatically by the simulation environment. The phase of generating equations from the description of connection between ports is referred to as connection semantics [4]. For example, the equations generated by the translator are listed in Figure 9 for a DC motor-load model.



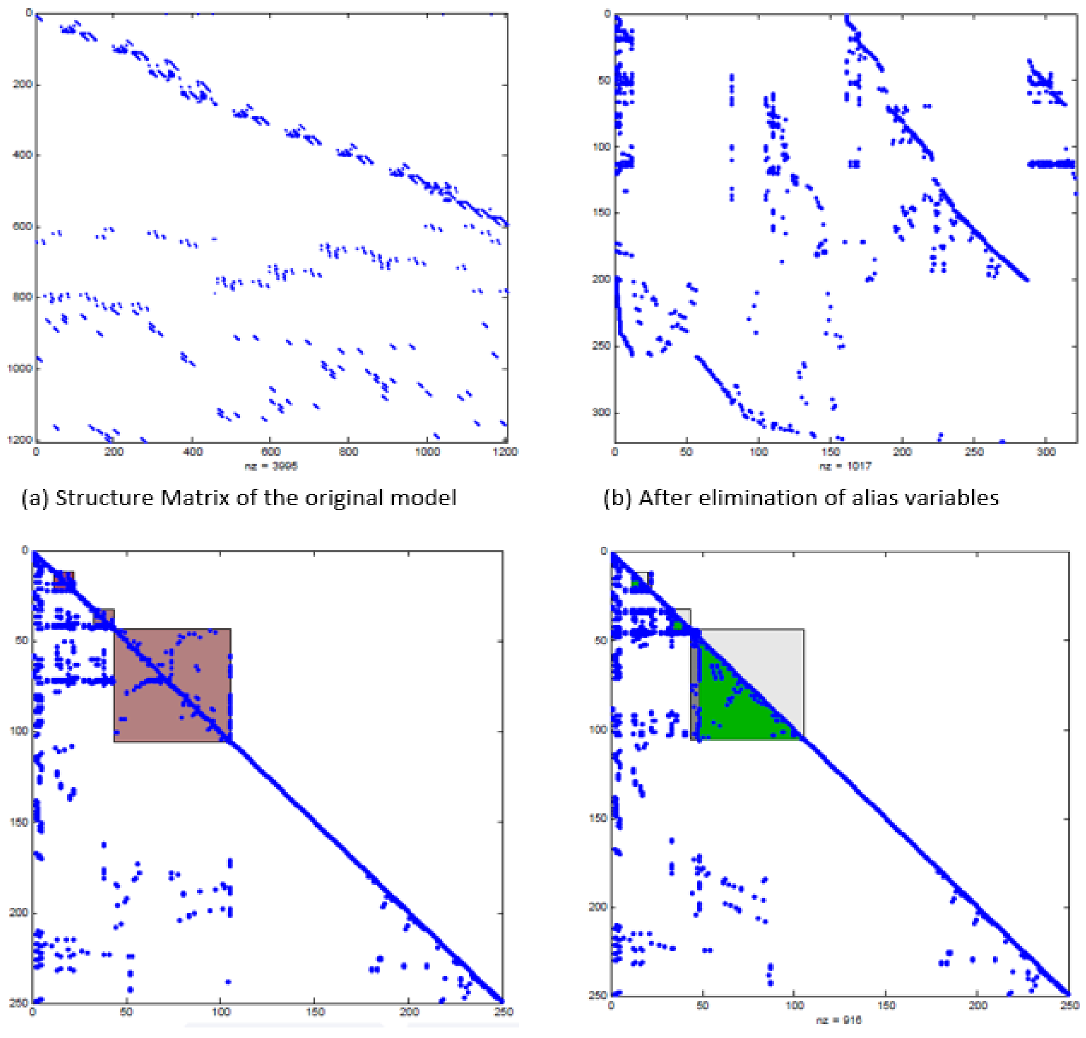
```
0 == DC.p.i + R.n.i     EM.u == EM.p.v - EM.n.v          R.u == R.p.v - R.n.v
DC.p.v == R.n.v         0 == EM.p.i + EM.n.i             0 == R.p.i + R.n.i
                        EM.i == EM.p.i                   R.i == R.p.i
0 == R.p.i + L.n.i      EM.u == EM.k * EM.ω              R.u == R.R * R.i
R.p.v == L.n.v          EM.i == EM.M / EM.k
                        EM.J * EM.ω == EM.M - EM.b * EM.ω   L.u == L.p.v - L.n.v
0 == L.p.i + EM.n.i                                      0 == L.p.i + L.n.i
L.p.v == EM.n.v         DC.u == DC.p.v - DC.n.v          L.i == L.p.i
                        0 == DC.p.i + DC.n.i             L.u == L.L * L.i'
0 == EM.p.i + DC.n.i    DC.i == DC.p.i
EM.p.v == DC.n.v        DC.u == DC.Amp * Sin[2 π DC.f * t]

0 == DC.n.i + G.p.i                (load component not included)
DC.n.v == G.p.v
```

Fig. 9 Equations derived automatically from the graphical model by Modelica [4]

If a DAE solver is used directly, the simulation will be very slow and initialization might not be possible for higher index systems (meaning number of states needed for the model, which essentially determine the number of independent initial conditions, are less than the number of differentiated variables). Therefore, the model equations are first transformed into a form that is better suited for numerical solvers during the "Translation" process, which is an ODE or index-1 DAE.

The following section attempts to give an overview of the steps involved in the same [4]:

- Formation of a Structure Matrix:
  From the model equations derived through the graphical layout of the model, a matrix that describes the dependency of each equation on its variables is created. The matrix results from each equation being assigned a row and each variable being assigned a column. If an equation makes use of a certain variable, the entry is assigned a value of 1, else the entry is 0 (see Eqn 36 or Figure 10(a)). The aim is to rearrange the Structure Matrix into a BLT (block lower triangle) so that each term can be solved for sequentially.



(a) Structure Matrix of the original model

(b) After elimination of alias variables

(c) BLT partitioning

(d) After tearing

Fig. 10 Structure matrix of a full-vehicle model [4]

- Constants and alias variables reduction:
  To improve the computational efficiency of the solver, the system of equations is simplified as much as. Two simplification methods are applied. Firstly, from examining the Structure matrix, it is possible to determine which variables are really constants. Secondly, by examining the rows that contain only two variables, it is possible to determine which variables may be alias (i.e. a = b or c = -b). Such variables are eliminated as is intuitive. This usually results in a significant reduction in complexity. For example, for a full-vehicle model [5], the number of unknowns is reduced from 1200 to 330 (see Figure 8(b)).

- DAE to ODE

  As mentioned before, DAE solvers are typically slow especially in terms of initialization, so the system is generally transformed into a set of ODEs, by differentiating the relevant equations analytically a specific number of times (Pantelides algorithm [7]). Selection of which variables to use as state variables (that generate minimal algebraic loops) is done statically during translation. For example, two rotating bodies with inertia J1 and J2 connected via a fixed gear ratio n between the two bodies (see Figure 11).

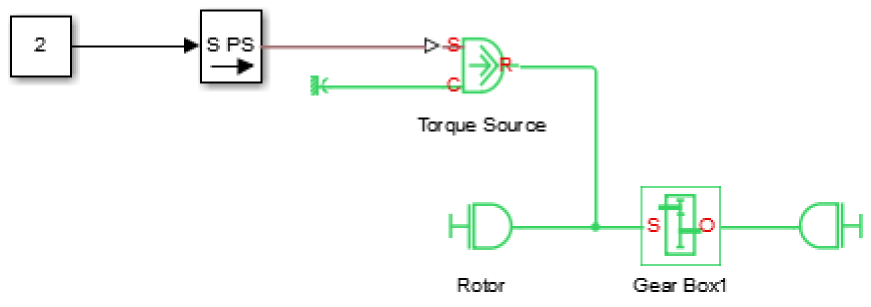


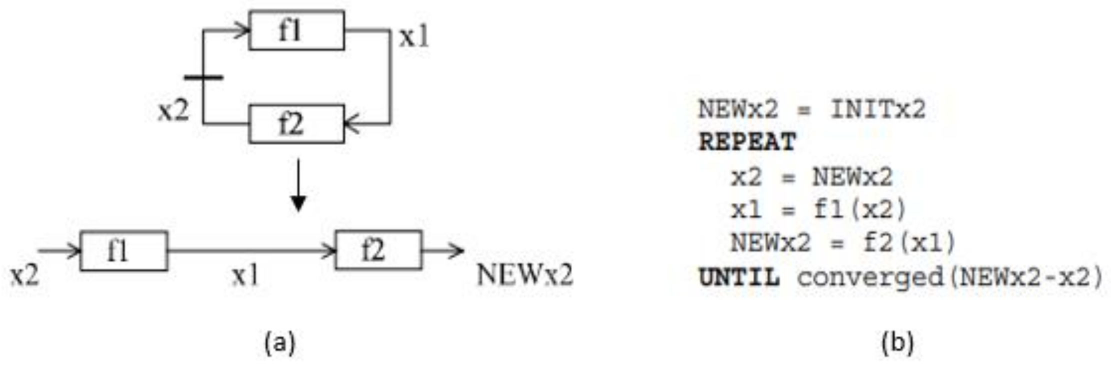
Fig. 11 A simple rotational mechanical system

The translation process would involve differentiating the position constraint twice to calculate the reaction torque in the coupling, and it is sufficient to select the velocity of either body as state variables. The constraint leads to a linear system of simultaneous equations (see Eqn 37) involving angular accelerations and torques.

$$\begin{bmatrix} J_1 & 0 \\ 0 & nJ_2 \end{bmatrix} \begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \qquad (37)$$

- Block Lower Triangle Partitioning:
  In most cases, after the aforementioned steps, the Structure Matrix can be arranged as a lower triangular matrix and variables can then be solved one by one. However, this might not be the case and algebraic loops or coupled equations will be present (see Figure 10(c)). The blocks along the diagonal of the BLT structure represent the identified algebraic loops (Tarjan's algorithm [10]). These algebraic loops are two-way coupled sets of equations, possibly, but not necessarily, differential equations, requiring simultaneous solution. A linear small algebraic loop is solved symbolically.

- Tearing:
  Tearing [9] is used to reduce the size of the algebraic loops. This method can be seen as breaking an algebraic loop by removing the algebraic connection between the output of the system and the input (see Figure 12) and compensating, by constraining the resulting system such that ‘x2’ must equal ‘NEWx2’. Iteration variables (called “tearing variables”) are selected, and a process of converging iterations is undertaken.



```
NEWx2 = INITx2
REPEAT
  x2 = NEWx2
  x1 = f1(x2)
  NEWx2 = f2(x1)
UNTIL converged(NEWx2-x2)
```

Fig. 12 ‘Tearing’ process to avoid algebraic loops

An example of the complete process as performed by the translator which essentially involves equation forming and partitioning is mentioned in the Appendix for an inverter model from the author of Dymola [5].

Thus, as seen that much of the work can be left to the modelling environment. The ability to formulate problems as DAEs rather than ODEs reduces the burden on the model developer because less effort is involved in formulating equations than to finding ways to make the model reusable. Example of tools include Dymola, Saber, and Simscape.

VII. Compatibility of an EEO tool with PhySiMod

The popular EEO language tools are Dymola and Simscape. Both can be readily integrated into the Simulink environment. As an example, Figure 13 shows a Battery Electric Vehicle plant model with the electric powertrain based in Simscape, and the vehicle dynamics based on PhySiMod.

Choice between Dymola and Simscape language is a tradeoff between using a tool with an open collaborative library and using an integrated tool-chain. In terms of license availability, Simscape is available as a toolbox and thus, a ready integration in the Matlab tool chain (Simulink/Stateflow/Simscape).

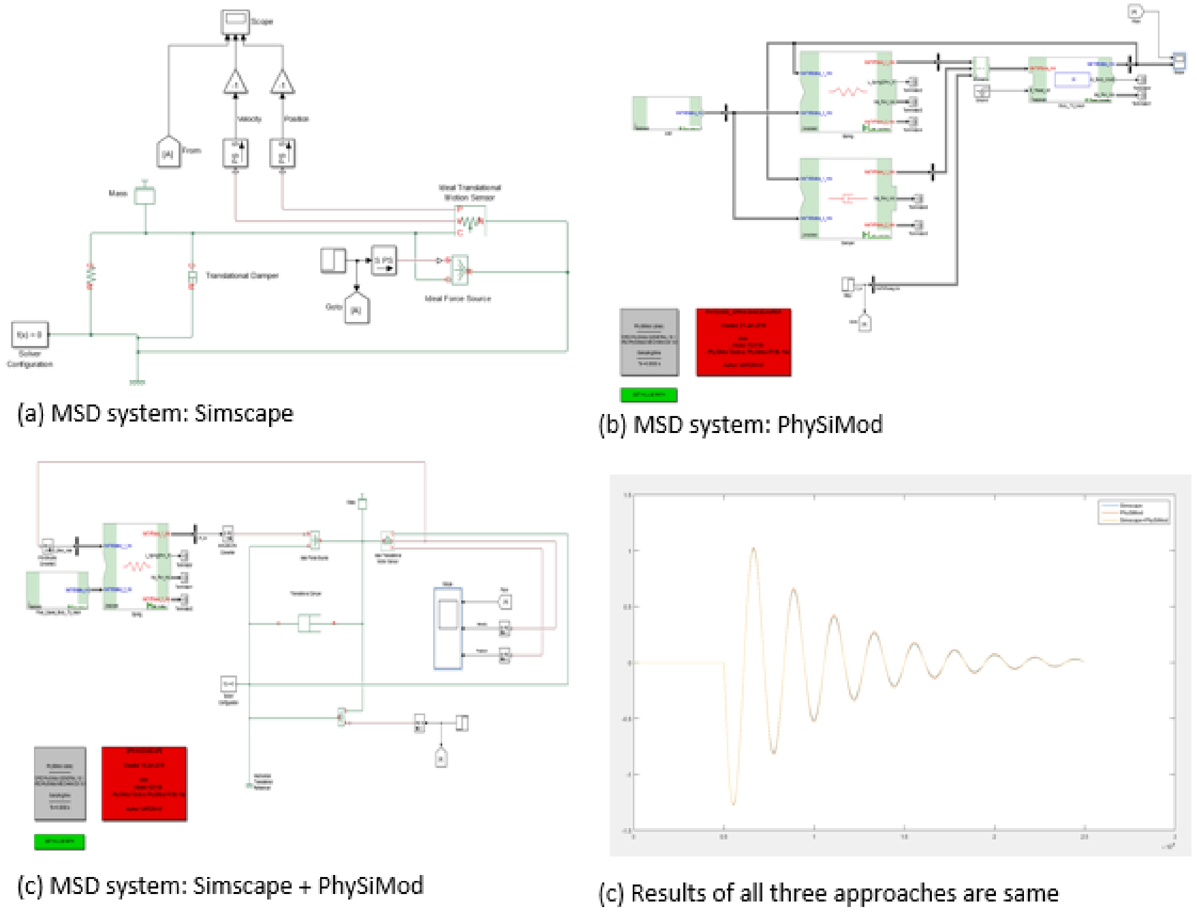

(a) MSD system: Simscape
(b) MSD system: PhySiMod
(c) MSD system: Simscape + PhySiMod
(c) Results of all three approaches are same

Fig. 13 Equivalency of Simscape and PhySiMod

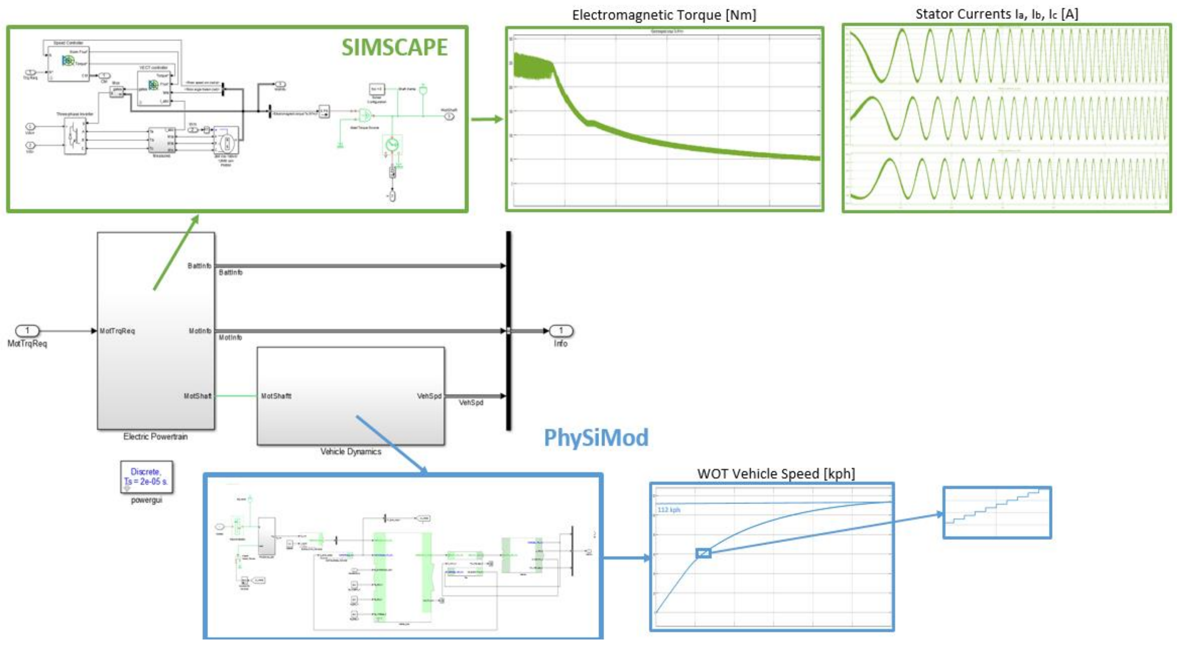


Fig. 14 A model of a Battery Electric Vehicle implemented using Simscape and PhySiMod

VIII. Advantages of Simscape

Direct integration of the workflow with the products of the Mathworks family, is the main advantage of Simscape. In terms of the library, Simscape consists of detailed and validated components which is regularly updated. In terms of modeling electric drivetrains, Simscape has two libraries for 3 phase electrical systems named SimPower Systems and Specialized Technology respectively as of Matlab 2015b.

SimPower Systems fully utilizes the Simscape technology and the component models are written in the Simscape language.

Specialized technology is optimized for modeling electrical power systems. It is mature and has been refined and tuned for more than a decade. The Specialized technology block libraries contain a large number of models that utilize their own electrical domain. Its primary advantage are that the outputs are Simulink signals.

Appendix II mentions an overview of the working of electric drivetrain components and the associated Simscape library block.

IX. Conclusion

Equation-based modelling facilitates easy modeling of physical systems in different domains. More complex system benefit more from taking advantage of the better structuring and increased possibility for reusability. Industry is moving away from modeling everything in Simulink to also used Modelica [6] and/or Simscape. Since Simscape is already available in the MathWorks tool-chain, adding Simscape capabilities to extend the existing PhySiMod library where needed should be considered.

References

[1] Bowles, P., Tiller, M., Elmqvist, H., Mattsson, S.E., and Otter, M. (2001): “Feasibility of detailed vehicle modeling”. Proc. 2001 SAE Congress and Exposition.

[2] Otter, M., C. Schelegel, and H. ElmqvistT (1997): “Modeling and realtime simulation of an automatic gearbox using Modelica.” Proc. European Simulation Symposium (ESS’97).

[3] Elmqvist H., Mattsson S.E. (1997): “An introduction to the physical modeling language Modelica”. Proc. ESS '97 European Simulation Symposium.

[4] Dassault Systems AB (2013), Dymola User Manual Volume 2.

[5] Elmqvist, H. (1978). A Structured Model Language for Large Continuous Systems Department of Automatic Control, Lund Institute of Technology (LTH).

[6] Elmqvist H., (2014): “Modelica Evolution – From My Perspective”. Proc. 10th International Modelica Conference.

[7] Pantelides C. (1988): The Consistent Initialization of Differential-Algebraic Systems, SIAM J. Sci. Stat. Comput., 9(2), pp. 213-231.

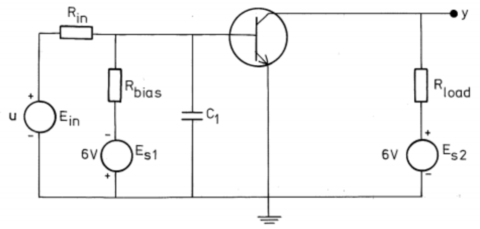
[8] Mattsson, S.E. and G. Söderlind (1993): Index reduction in differential-algebraic equations using dummy derivatives. SIAM Journal of Scientific and Statistical Computing, Vol. 14 pp. 677 - 692, 1993.

[9] Elmqvist, H., and M. Otter (1994): Methods for Tearing Systems of Equations in Object Oriented Modeling, Proc. ESM'94, European Simulation Multiconference, Barcelona, Spain, June 1- 3, 1994, pp. 326-332.

[10] R. Tarjan. Depth-first search and linear graph algorithms. SIAM Journal on Computing, 1(2):146–160, 1972.

APPENDIX I

The translation process as implemented by Dymola for an inverter model [5]:



```
{ Interaction with the translator }

>@add ellib
>@add inv
>print equations

 Tr::Rbb        V = Va - Vb
                R*I = V
 C1             V = Va - Vb
                C*derV = I
 Ein            V = Vb - Va
 Common         V = 0
 Tr::Cemit      V = Va - Vb
                Q = C*V
                derQ = I
 Tr::Diode1     V = Va - Vb
                I = I0*(exp(K*V) - 1)
 Tr::Diode2     V = Va - Vb
                I = I0*(exp(K*V) - 1)
 Tr::Ccoll      V = Va - Vb
                Q = C*V
                derQ = I
 Tr             Cemit.C = 3.0E-12 + 6.7E-9*Diode1.I
                Ccoll.C = 2.0E-12 + 180.0E-9*Diode2.I
                Revcurr.I = A1*Diode2.I
                Forwcurr.I = A2*Diode1.I
                Diode1.Vb = Ve
                Cemit.Va = Diode1.Vb
                Cemit.I + Revcurr.I = Ie + Diode1.I
                Diode2.Vb = Vc
                Ccoll.Va = Diode2.Vb
                Ccoll.I + Forwcurr.I = Ic + Diode2.I
                Rbb.Va = Vb
                Rbb.I = Ib
                Diode2.Va = Rbb.Vb
                Ccoll.Vb = Diode2.Va
                Diode1.Va = Ccoll.Vb
                Cemit.Vb = Diode1.Va
                Diode2.I + Diode1.I = Rbb.I + Ccoll.I +
                  Forwcurr.I + Cemit.I + Revcurr.I
```

```
 Es1             V = Vb - Va
 Es2             V = Vb - Va
 Rin             V = Va - Vb
                 R*I = V
 Rbias           V = Va - Vb
                 R*I = V
 Rload           V = Va - Vb
                 R*I = V
 inv             Ein.V = U
                 Y = Rload.Vb
                 Es1.V = 6
                 Es2.V = 6
                 Rin.Va = Ein.Vb
                 Rin.I = Ein.I
                 Es1.Va = Rbias.Vb
                 Es1.I = Rbias.I
                 C1.Vb = Es1.Vb
                 Tr.Ve = C1.Vb
```

```
                   Common.V = Tr.Ve
                   Ein.Va = Common.V
                   Es2.Va = Ein.Va
                   Rbias.Va = Rin.Vb
                   Cl.Va = Rbias.Va
                   Tr.Vb = Cl.Va
                   Rbias.I + Cl.I + Tr.Ib = Rin.I
                   Rload.Va = Es2.Vb
                   Rload.I = Es2.I
                   Tr.Vc = Rload.Vb
                   Tr.Ic = Rload.I

>partition
>print solved

 Common            V = Ø
 inv               Tr.Ve = Common.V
                   Cl.Vb = Tr.Ve
 Cl                Va = V + Vb
 inv               Tr.Vb = Cl.Va
 Tr                Rbb.Va = Vb
                   Diodel.Vb = Ve
                   Cemit.Va = Diodel.Vb

-Tr::Diodel        V = [Va] - Vb
-                  I = IØ*(exp(K*[V]) - 1)
-Tr                Cemit.C = 3.ØE-12 + 6.7E-9*[Diodel.I]
-Tr::Cemit         Q = [C]*V
-                  [V] = Va - Vb
-Tr                [Cemit.Vb] = Diodel.Va

                   Ccoll.Vb = Diodel.Va
                   Diode2.Va = Ccoll.Vb
                   Rbb.Vb = Diode2.Va
 Tr::Rbb           V = Va - Vb
                   I = V/R
 inv               Rbias.Va = Cl.Va
                   Esl.V = 6
                   Esl.Vb = Cl.Vb
 Esl               Va = Vb - V
 inv               Rbias.Vb = Esl.Va
 Rbias             V = Va - Vb
                   I = V/R
 Tr                Ib = Rbb.I
 inv               Ein.V = U
                   Ein.Va = Common.V
 Ein               Vb = V + Va
 inv               Rin.Va = Ein.Vb
                   Rin.Vb = Rbias.Va
 Rin               V = Va - Vb
                   I = V/R
 inv               Cl.I = Rin.I - (Rbias.I + Tr.Ib)
 Cl                derV = I/C
 inv               Ein.I = Rin.I

-Tr::Diode2        [I] = IØ*(exp(K*V) - 1)
-                  [V] = Va - Vb
-Tr                Ccoll.Va = [Diode2.Vb]
```

```
 -Tr::Ccoll      V = [Va] - Vb
 -               Q = C*[V]
 -Tr             [Ccoll.C] = 2.ØE-12 + 18Ø.ØE-9*Diode2.I

                 Revcurr.I = A1*Diode2.I
                 Forwcurr.I = A2*Diodel.I
  inv            Es2.V = 6
                 Es2.Va = Ein.Va
  Es2            Vb = V + Va
  inv            Rload.Va = Es2.Vb
  Tr             Vc = Diode2.Vb
  inv            Rload.Vb = Tr.Vc
  Rload          V = Va - Vb
                 I = V/R
  inv            Tr.Ic = Rload.I
  Tr             Ccoll.I = Ic + Diode2.I - Forwcurr.I
                 Cemit.I = Diode2.I + Diodel.I - (Rbb.I +
                   Ccoll.I + Forwcurr.I + Revcurr.I)
  Tr::Cemit      derQ = I
  Tr::Ccoll      derQ = I
  Tr             Ie = Cemit.I + Revcurr.I - Diodel.I
  inv            Esl.I = Rbias.I
                 Es2.I = Rload.I
                 Y = Rload.Vb
```

APPENDIX II

In the following section, basics of a motor drive is overviewed. Firstly, inverter and its modulation techniques are reviewed. Then motor types and motor control relevant to xEVs are reviewed. Following this related Simscape library blocks are discussed with respect to their inputs, outputs and parameters.

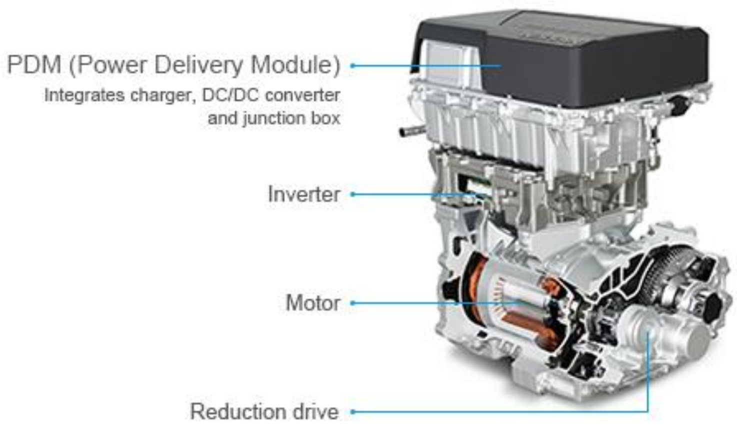

PDM (Power Delivery Module)
Integrates charger, DC/DC converter and junction box
Inverter
Motor
Reduction drive

Fig. A.2.0 A typical motor drive

Inverter:

AC motors are fed by a variable AC voltage and frequency produced by an inverter. PWM Modulation techniques are implemented in an inverter (see Figure A.2.1) to convert the DC bus voltage to AC with desired amplitude and frequency for the motor. There are different types of modulation techniques, but the most used ones are:

- Sine-triangle PWM
- Hysteresis modulation
- Space Vector Modulation

The basic idea behind the three techniques is the same, i.e. to control switches (turn them on-off suitably) to generate a reference signal (a sine-wave). This switch on-off frequency is made to be much higher than (greater than 10 times in general) to approximate the reference signal well. Each of the 3 techniques are briefly discussed.

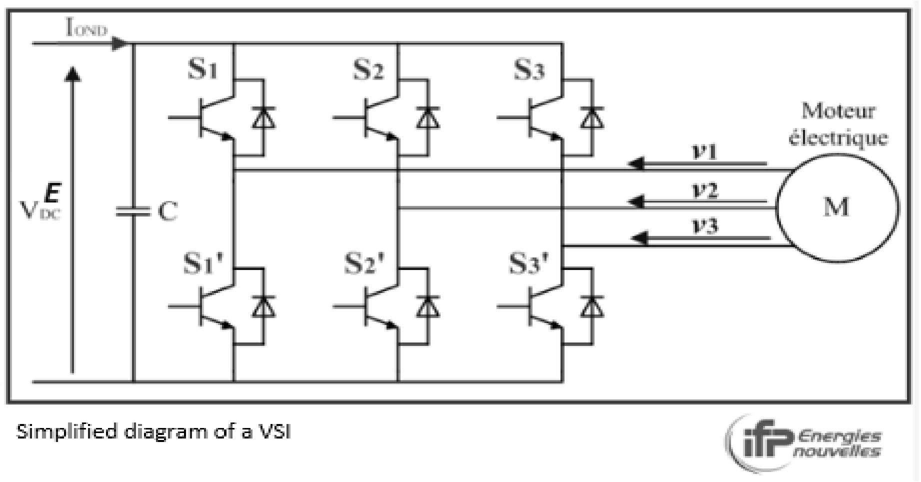


Fig A.2.1 An inverter schematic

Sine-triangle Modulation:

For now, just consider a single arm (S1 and S1' in Figure A.2.2) of the inverter to generate a single sinusoidal waveform (Note that a full-inverter would generate 3 phase-shifted sine waveforms).

Based on Pulse-Width Modulation (PWM) concept, if the duty-cycle (see Figure A.2.2) is changed sinusoidally, a sinusoidal voltage will be generated at the output. Then, one can realize the need to have a rule to make the switches behave in that manner. One way (as is done in Sine-triangle Modulation), is to take a triangle signal (called carrier signal) with much higher frequency (>10 times) than the reference signal. Its frequency is the PWM frequency. These two signals are compared (see Figure A.2.3). At the time when the reference signal is larger than the triangle signal, the upper switch is turned on and the lower switch is off, otherwise, the upper switch is off and the lower switch is on. The resultant waveform approximates a sinusoidal one with high enough switching frequency (see Figure A.2.4).

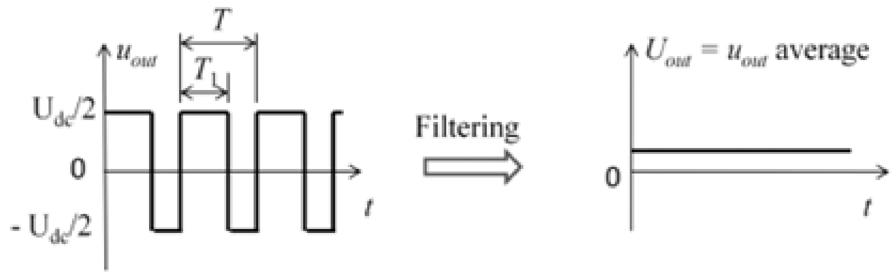


Fig A.2.2 Pulse-width modulation

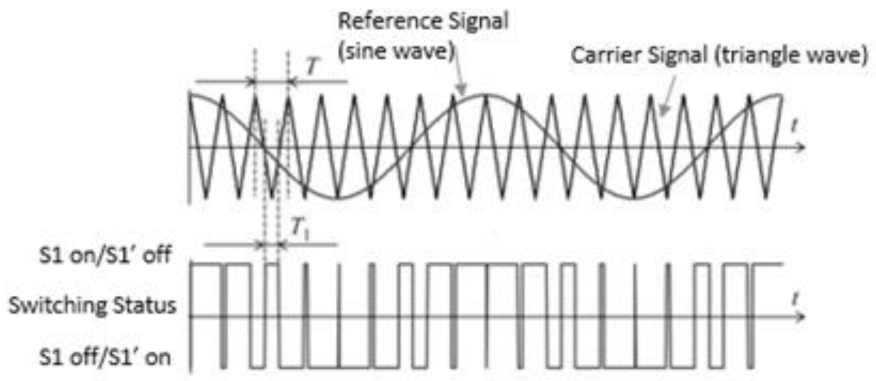


Fig A.2.3 Sine-triangle modulation

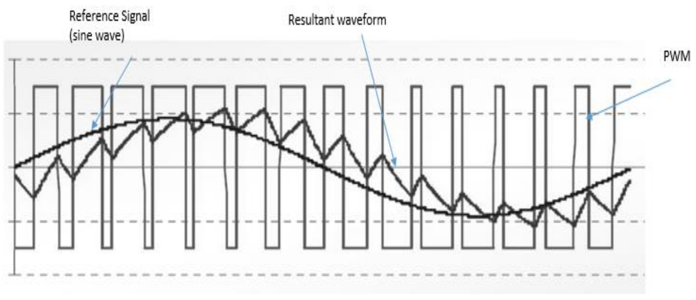


Fig. A.2.4 Resultant sinusoidal waveform from sine-triangle modulation

Hysteresis Modulation:

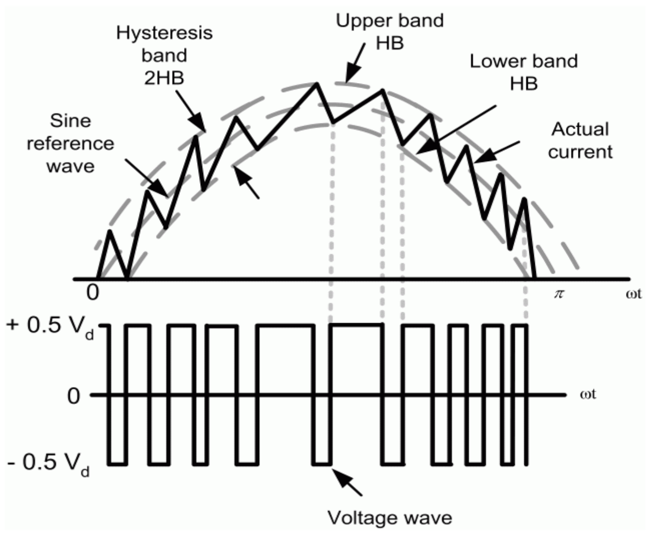
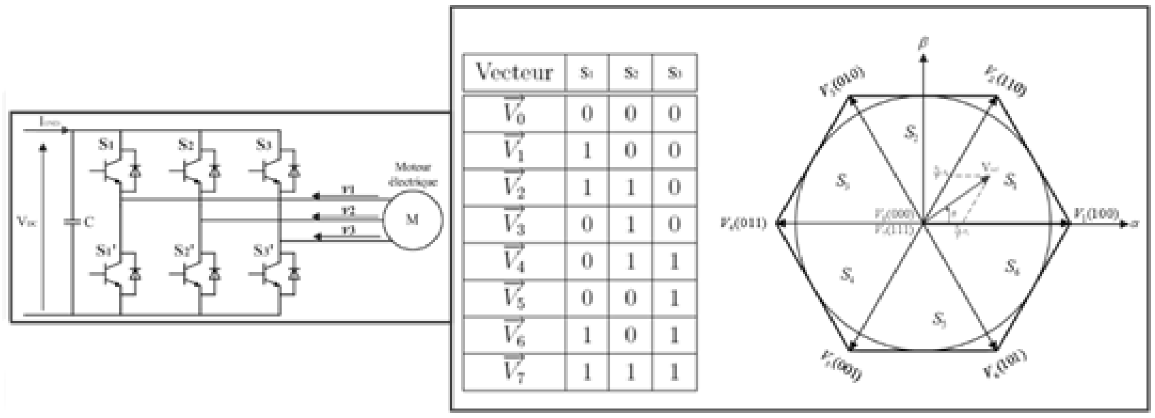


Fig. A.2.5 Hysteresis modulation

In this method, the reference sinusoidal waveform is compared to the actual motor line current. A band is defined for the reference wave. For the rule when to switch on or off, if the motor line current exceeds the upper limit of the band, the upper switch of the inverter arm (S1 for our consideration) is turned off and the lower switch (S1') is turned on. The current starts to decay as a result and the opposite operation is performed when the current reaches the lower limit of the band (see Figure A.2.5).

Space-Vector Modulation:



Vecteur	S1	S2	S3
$\overrightarrow{V_0}$	0	0	0
$\overrightarrow{V_1}$	1	0	0
$\overrightarrow{V_2}$	1	1	0
$\overrightarrow{V_3}$	0	1	0
$\overrightarrow{V_4}$	0	1	1
$\overrightarrow{V_5}$	0	0	1
$\overrightarrow{V_6}$	1	0	1
$\overrightarrow{V_7}$	1	1	1

Fig. A.2.6 SVM

The space vector modulation technique differs from the previous techniques in that there are no separate comparisons used for each of the three phases. Instead, a reference voltage space vector is produced as a whole.

There are 8 switch states. The output voltages of the inverter are composed by these 8 switch states. We can define 8 voltage vectors corresponding to each switch state. These 8 voltage vectors form the voltage-vector space divided into 6 sectors (see Figure A.2.6).

Now, it is known that a 3-phase AC signal can be represented by an equivalent 2-phase signal by suitable transformation. The three phases set-point voltages are represented by a single vector in the 2-phase plane. This vector is then approximated by (over the modulation period) by a combination of vectors representing the various operating states of the inverter.

Looking down the columns for the active switching vectors V1-6, the output voltages vary as a pulsed sinusoid, with each leg offset by 120 degrees of phase angle.

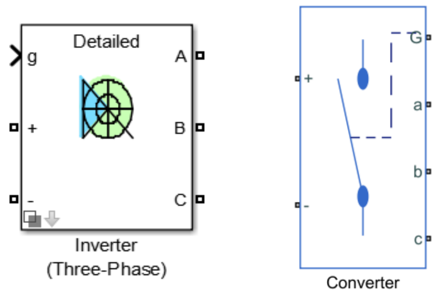


Fig. A.2.7. Simscape implementation of inverter

The inverter used in the AC drive models of the ‘Specialized Technology’ library is based on two types of modulation, hysteresis modulation and space vector pulse width modulation (PWM).

Motors:

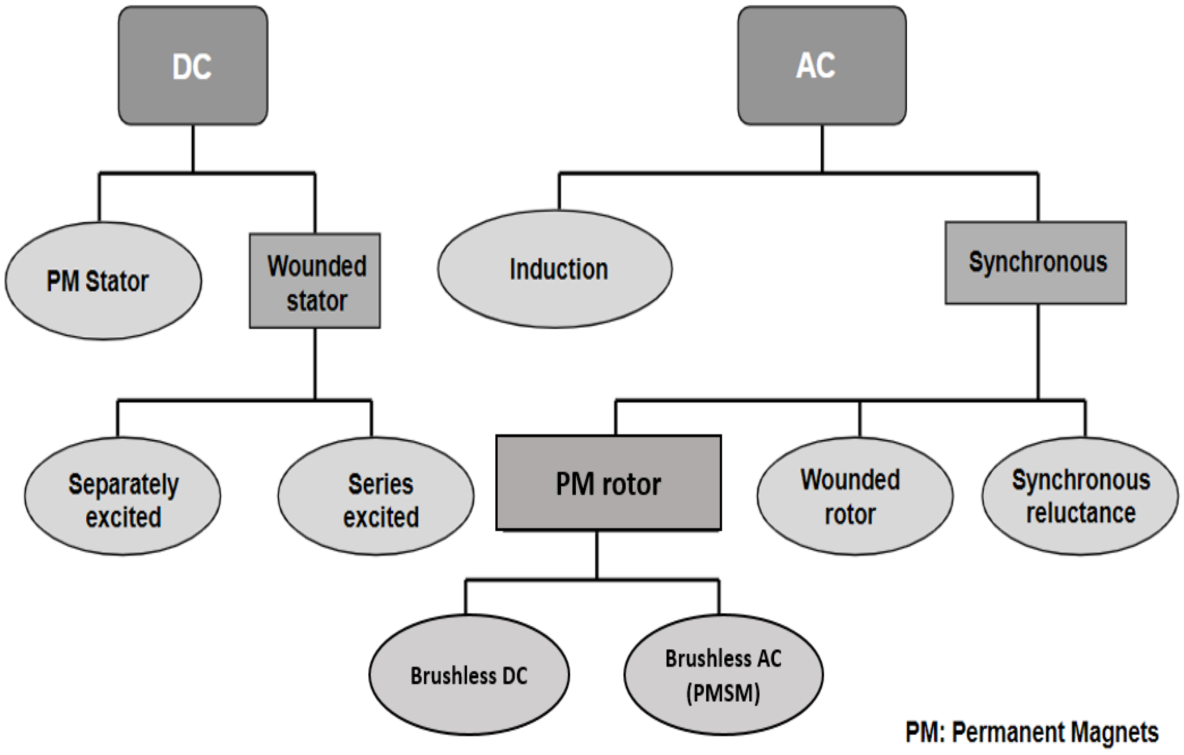


Fig. A.2.8 Types of motors

The following table depicts the type of motors used in some of the popular Electric/Hybrid Vehicles.

Vehicle Name	Motor Type
GM EV1	Induction Motor
Ford Think City	Induction Motor
Hyundai Kona	PMSM
Tesla Model S, X	Induction Motor
Tesla Model 3	PMSM
Chevy Bolt	PMSM
Toyota Prius	PMSM

As is clear, most use either Induction Motors (Asynchronous) or Permanent-Magnet Synchronous Motors (PMSM). Therefore, the following two are discussed although there are many other types too (see Figure A.2.8).

Rotating magnetic field:

The main characteristic of rotating electric machines is the production of a rotating magnetic field through a 3 phase AC supply. It is important to realize this fact. To put simple, the stator windings of a motor comprises of 3 windings (a-b-c). Consider one of those coils (coil-a), to which a sinusoidal voltage is applied. This would produce a sinusoidal, pulsating magnetic field in the winding (see Figure A.2.8).

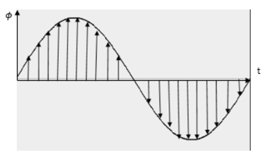


Figure A.2.9 A single phase AC applied to a single coil creates a pulsating field

Now, if 2 more phase shifted sinusoidal voltage are applied to 2 more such winding placed coaxially but physically phased too, then at time t, the field vectors from the 3 windings can be added to give a net field vector. With evolution of time, the net effect is to obtain a resultant rotating field vector (see Figure A.2.10).

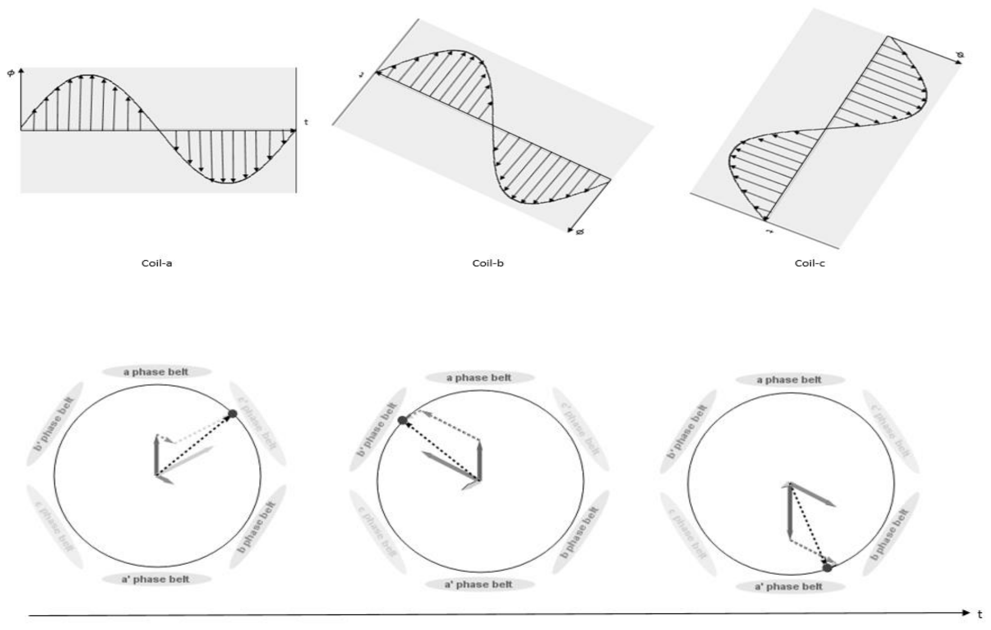


Figure A.2.10 A 3 phase AC applied to three coils phase shifted produces the effect of a rotating magnetic field

Another important aspect is to realize that all rotating electric machines are essentially magnetically coupled circuits, and therefore, their equivalent circuit models include flux linkage terms (self and mutual) which can be expressed in terms of inductances. Therefore, in general, input to such models are equivalent winding resistances and inductances.

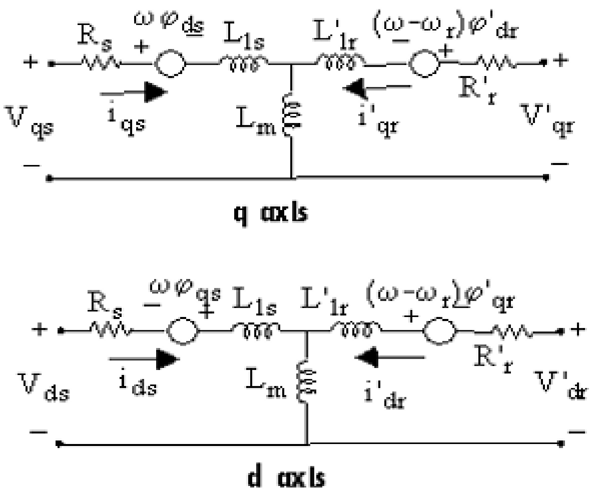


Figure A.2.11 Equivalent circuit model of electric machines

Induction motor (Asynchronous motor):

In induction motors, a 3 phase supply to the stator windings creates a rotating magnetic field as previously discussed. This field induces current in the rotor which is either wound-type (i.e. made up of coils) or squirrel-cage type (made up of copper/aluminum bars). The induced currents create a field of their own in order to minimize the changing flux through the rotor, thus causing the rotation of the rotor asynchronously.

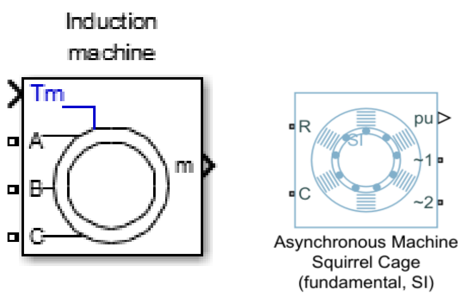
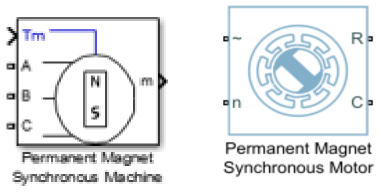


Fig. A.2.12 Simscape implementation of IM

PMSM motor

In PMSM, the electromagnetic torque is a result of interaction between the rotating magnetic field, produced in the stator windings supplied by 3-phase AC, and that produced in the rotor by permanent magnets (neodymium). The speed of rotation the rotor is equal to the speed of rotation of the stator rotating field (synchronous).

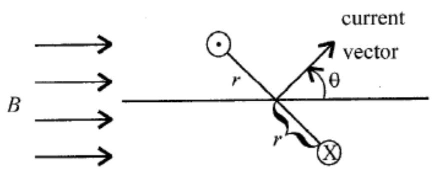


Tm
A
B
C
N
S
m
Permanent Magnet Synchronous Machine
~
n
R
C
Permanent Magnet Synchronous Motor

Fig. A.2.13 Simscape implementation of PMSM

Motor control:

It is known that in separately excited DC machine, the field flux (stator flux for AC machines in general) and armature flux (rotor flux for AC machines in general) are independently supplied. This leads to independent control of the machine torque (rotor current) and machine flux (stator current). Hence, the control to achieve desired operating characteristics becomes much simpler.



Now, the basic premise of FOC can be understood by considering a current carrying loop in a magnetic field as shown. From Lorrentz law, it is known that

$$T = -2BiNLr\sin\theta$$

This shows that the magnitude of the torque is maximized when the current vector is perpendicular to the field flux vector. Same conclusion is readily applied to a 3 phase rotating machine.

Now, in a three phase system, when using Park transformation, one can transform abc values to dq axis (2 axis), in which d and q variables are constant in steady state. This is the main benefit of dq system, assuming that the angular rotation speed is known, and rotating dq system can be aligned in any direction. By this, machine model reduces to the model of DC machine. In general, align dq system to rotor flux (in IM drives), or to magnet flux (in PMSM drives).

Having the previous discussion in mind, thus, it is in ones' interest to have the ratio of currents in each phase (a-b-c) of the stator winding such that the resultant current vector is perpendicular to the rotor flux vector (from magnets in case of PMSM).

Such a control scheme is called flux-oriented control or vector control which is most widely used type in xEVs. Vector control is applicable to both induction and synchronous motors. Another motor control technique popular in the literature is direct torque control.

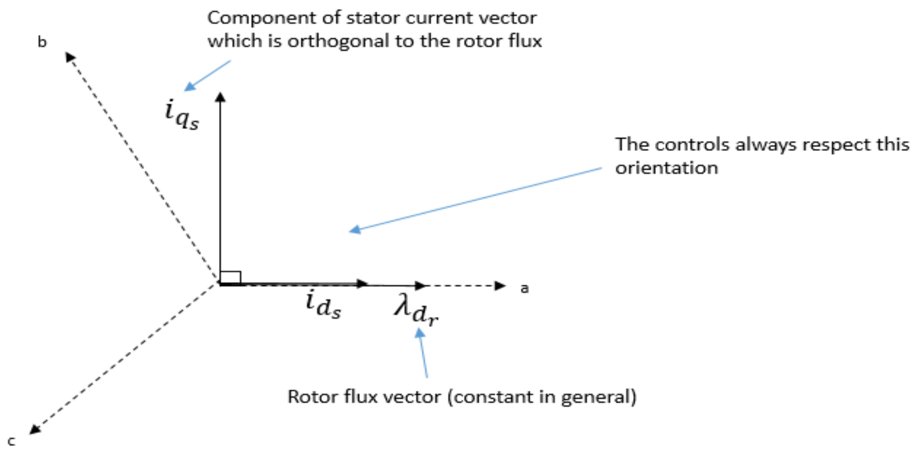

Component of stator current vector which is orthogonal to the rotor flux
b
i_{q_s}
The controls always respect this orientation
i_{d_s}
λ_{d_r}
a
Rotor flux vector (constant in general)
c

Fig. A.2.14 Flux-oriented control principle

Field-oriented Control:

The field-oriented control implies that the i_{ds} component of the stator current would be aligned with the rotor field and the i_{qs} component would be perpendicular to i_{ds}. The following relation holds for a reference frame rotating with the rotor flux (see Figure A.2.14):

$$T_e = 1.5p\frac{L_m}{L_r}(\phi_r i_{q_s})sin\theta,\ \theta = \pi/2$$

The analogy with DC machine performance should now be clear. The electric torque is proportional to the i_{qs} component whose amplitude is varied to get the desired torque.

A typical digital processor implementation would be follows:

I. Interrupt
II. Measure the stator phase current i_a , i_b and calculate i_c. This gives the position of the stator current vector.
III. Find the desired position of i_{q_s}. Since it should be orthogonal to the field vector, this can be found out by figuring out the position of rotor to get the position of the field vector:
    - through rotary encoders (PMSM)
    - through flux estimators (squirrel cage rotor IM)
IV. Once we know the desired position of i_{q_s}, an error value can be generated to regulate the actual stator phase currents. This is done by applying a Forward Clark-Park transform to get equivalent current values, which are DC in the synchronously rotating reference frame (see Figure A.2.15). These DC values are easier to process for the error to get the desired phase currents. Once the correction values are calculated, an Inverse Clark-Park transform is applied to get the individual values of the correction phase voltages which is the input to a PWM modulator.
V. Interrupt (after 10 us)
VI. Repeat

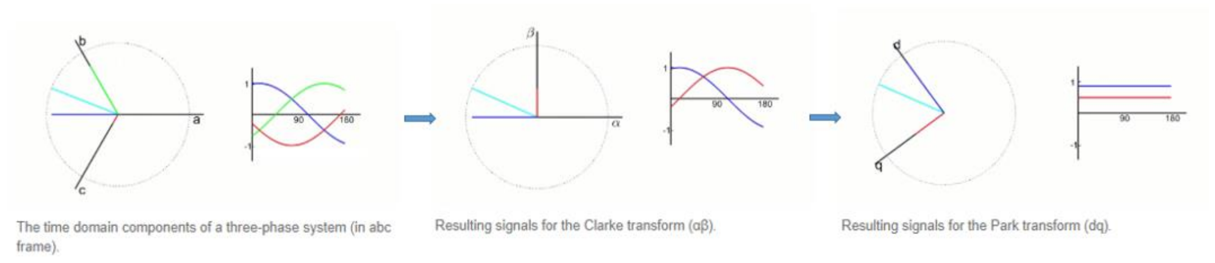

b
a
c
1
90
180
-1
The time domain components of a three-phase system (in abc frame).
β
α
Resulting signals for the Clarke transform (αβ).
d
q
Resulting signals for the Park transform (dq).

Fig. A.2.15 Clark-Park transform visualization

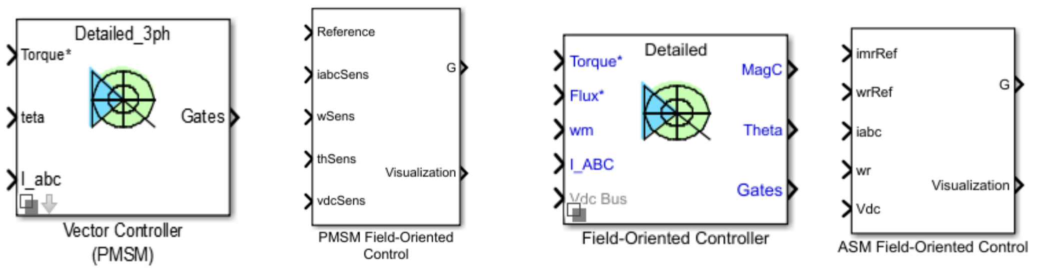


Fig. A.2.16 Simscape implementation of FOC/Vector control

Direct Torque Control:

This control method consists first in estimating the machine stator flux and electric torque in the stationary reference frame from terminal measurements. The estimated stator flux and electric torque are then controlled directly by comparing them with their respective demanded values using hysteresis regulators. The outputs of the two comparators are then used as input signals of an optimal switching table. This avoids the need of modulation unit and position sensors. Major disadvantage is due to the fact that switching frequency is highly variable, which leads to harmonic distortions and noise problems.

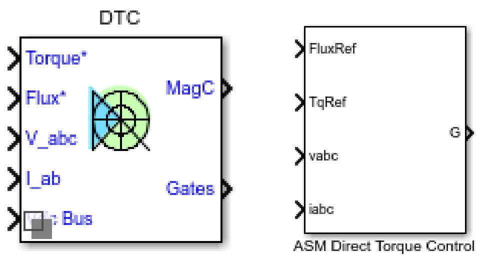


Fig. A.2.17 Simscape implementation of DTC

Battery:

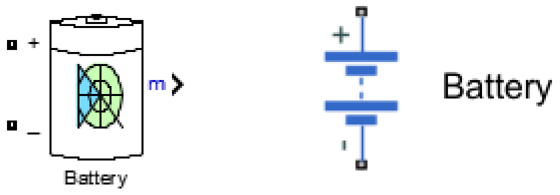


Fig. A.2.18 Simscape implementation of battery

To model a series and/or parallel combination of cells based on the parameters of a single cell, can be used.

Motor Drive:

Putting it all together, in the following section, a PMSM Drive and an Induction Motor Drive is put together based on the previously discussed blocks in Simscape.

The role of the speed controller is to keep the motor speed equal to the speed reference input in steady state and to provide a good dynamic during transients. It is a proportional-integral type controller. The motor speed ω is compared to the reference ω* and the error is processed by the speed controller to produce a torque command Te*.

The field-oriented control is modeled by the Vector Control block. The i_{qs}* and i_{ds}* current references are converted into phase current references i_a*, i_b*, i_c* for the current regulators. The regulators process the measured and reference currents to produce the inverter gating signals which is the input to the inverter model.

The induction motor is fed by the current-controlled PWM inverter, which operates as a three-phase sinusoidal current source.

For PMSM drive, the angle conversion block computes the electrical rotor angle from the mechanical rotor angle. Whereas IM drive estimates the motor's rotor flux.

PMSM Drive:

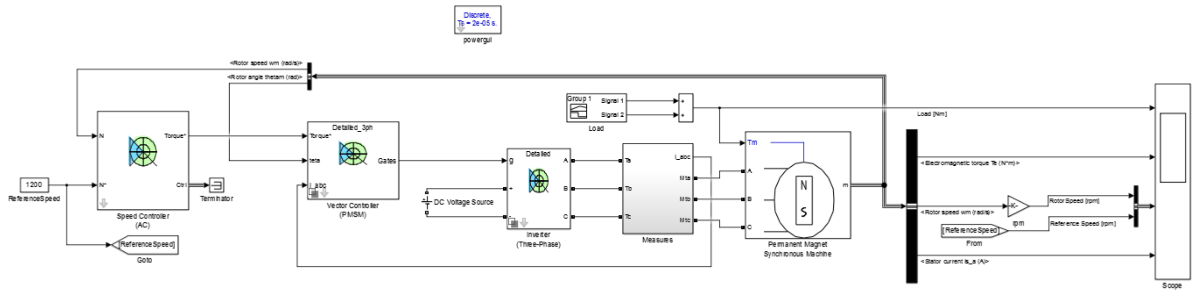


Fig. A.2.19 Simscape Specialized Technology implementation of PMSM Drive

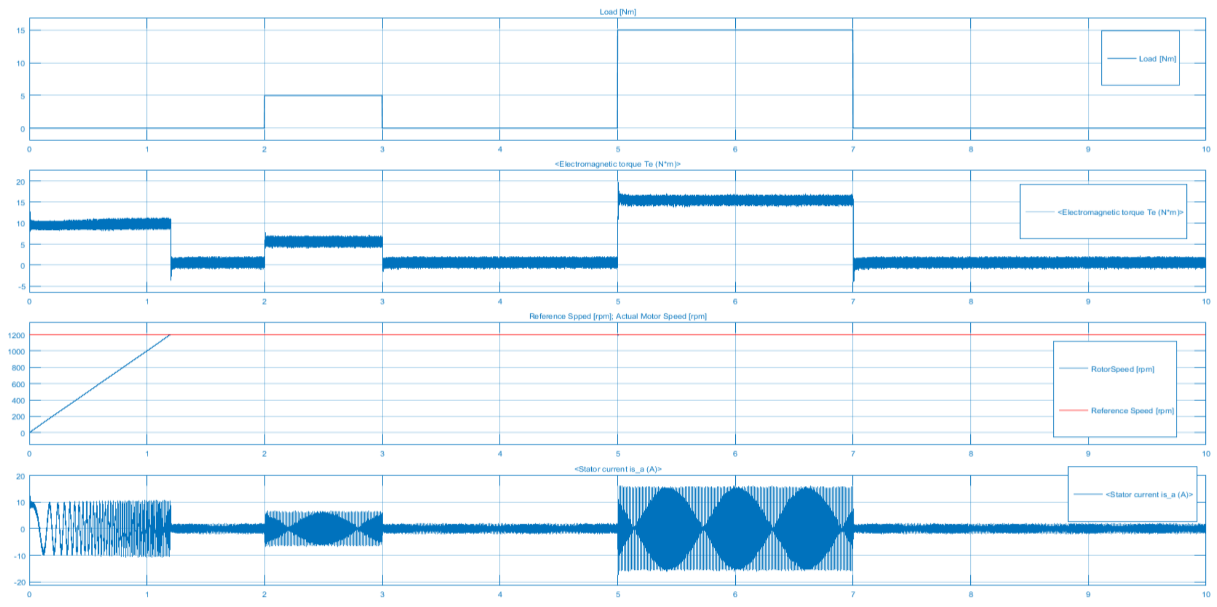


Fig. A.2.20 Results of PMSM Drive

Induction Motor Drive:

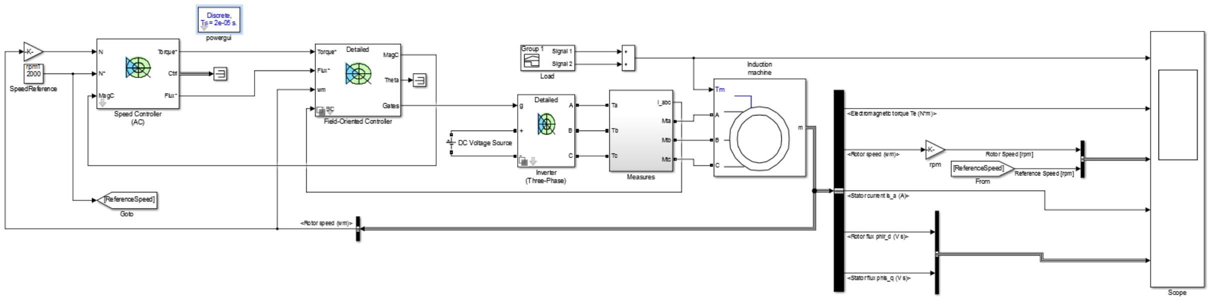


Fig. A.2.21 Simscape Specialized Technology implementation of IM drive

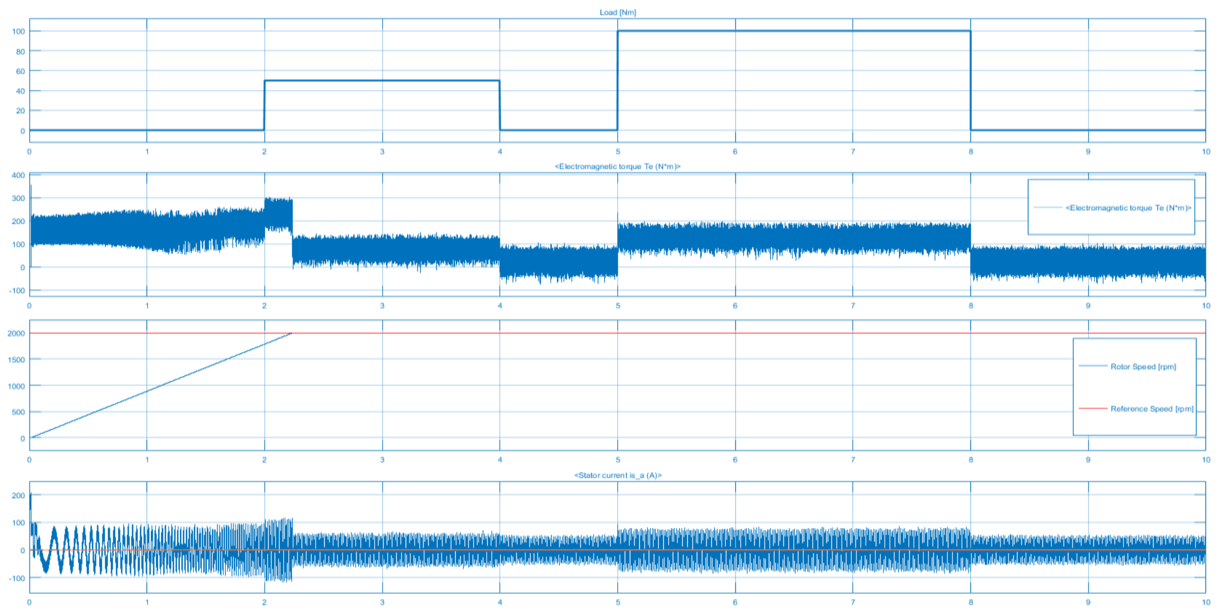


Fig. A.2.22 IM Drive results

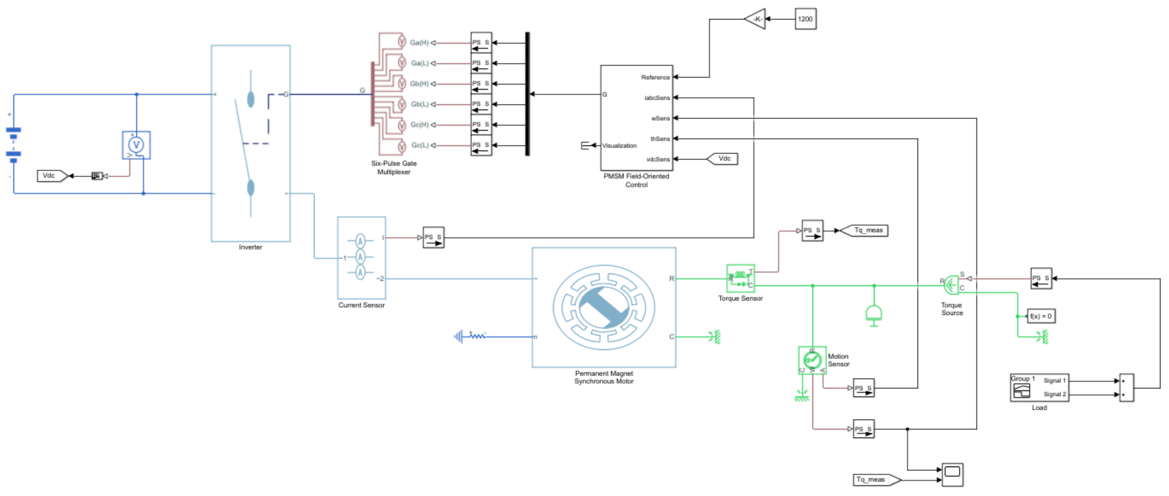


Fig. A.2.23 Simscape component implementation of a PMSM drive

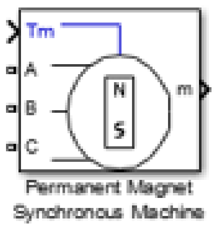
Simscape Library:

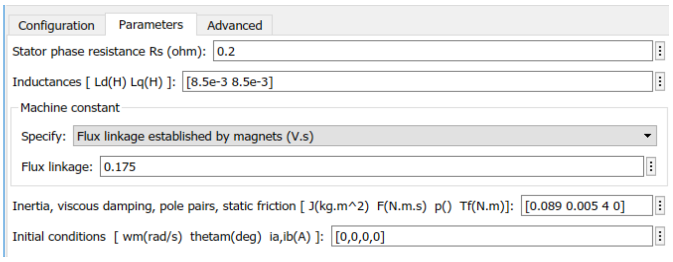
The simscape library for electric power systems has two types:

- Simscape Components
- Specialized Technology

Specialized Technology:

I. PMSM Motor block





Configuration | Parameters | Advanced

Stator phase resistance Rs (ohm): 0.2

Inductances [Ld(H) Lq(H)]: [8.5e-3 8.5e-3]

Machine constant

Specify: Flux linkage established by magnets (V.s)

Flux linkage: 0.175

Inertia, viscous damping, pole pairs, static friction [J(kg.m^2) F(N.m.s) p() Tf(N.m)]: [0.089 0.005 4 0]

Initial conditions [wm(rad/s) thetam(deg) ia,ib(A)]: [0,0,0,0]

Inputs/Outputs:

Tm

The Simulink input is the mechanical torque at the machine shaft.

A, B, C:

The 3 phase signal from inverter block

m

The Simulink output of the block is a vector containing measurement signals.

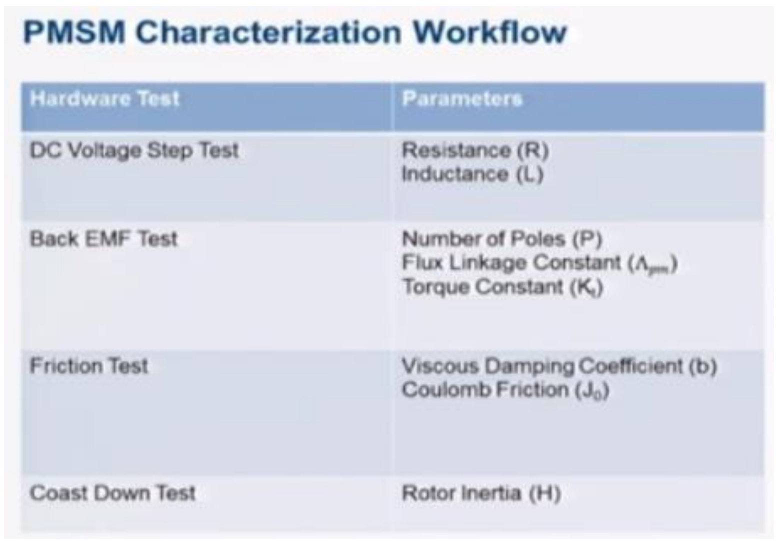
Parameters:

Electrical:

Parameter	Dimension	Description
R_s	(Ω)	Resistance one of the motor phase
L_d	(H)	d-axis inductance of one motor phase
L_q	(H)	q-axis inductance of one motor phase

Mechanical:

Parameters	Dimension	Description
J	(kg.m^2)	Total mechanical inertia
B_m	(N.m.s)	Viscous friction coefficient

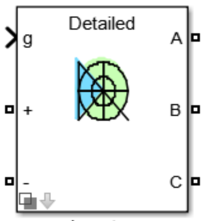


PMSM Characterization Workflow

Hardware Test	Parameters
DC Voltage Step Test	Resistance (R) Inductance (L)
Back EMF Test	Number of Poles (P) Flux Linkage Constant (Λ_{pm}) Torque Constant (K_t)
Friction Test	Viscous Damping Coefficient (b) Coulomb Friction (J_0)
Coast Down Test	Rotor Inertia (H)

Fig. A.2.24 Various tests required for parameters of an electric machine

II. Inverter block



Inverter
(Three-Phase)

<u>Inputs/Outputs:</u>

g

The gate input for the controlled switch devices. Pulses are sent to upper and lower switches of inverter legs A, B, and C.

+

The positive terminal on the DC side.

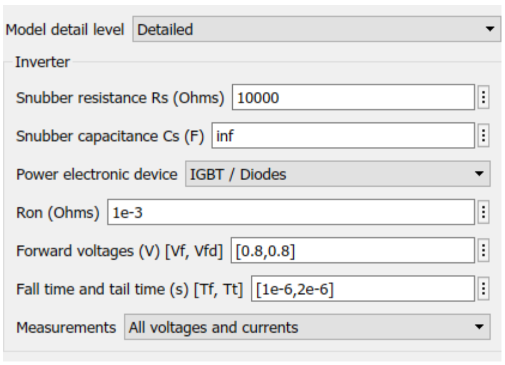
-

The negative terminal on the DC side.

A, B, C

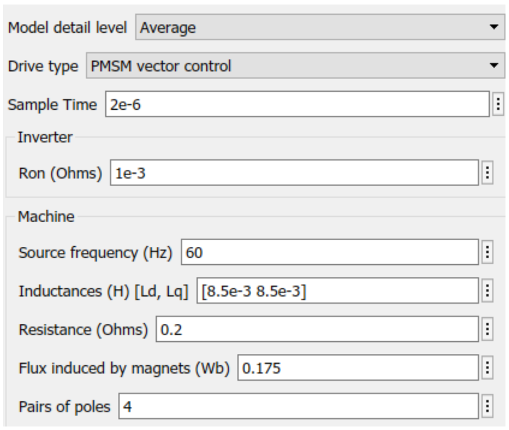
The three-phase terminals on the AC side.

<u>Parameters:</u>



Model detail level	Detailed
Inverter	
Snubber resistance Rs (Ohms)	10000
Snubber capacitance Cs (F)	inf
Power electronic device	IGBT / Diodes
Ron (Ohms)	1e-3
Forward voltages (V) [Vf, Vfd]	[0.8,0.8]
Fall time and tail time (s) [Tf, Tt]	[1e-6,2e-6]
Measurements	All voltages and currents

In detailed mode, the only important parameters are Ron and Forward voltages which might differ depending on the inverter.



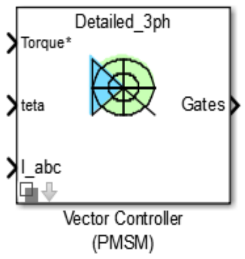
Model detail level	Average
Drive type	PMSM vector control
Sample Time	2e-6
Inverter	
Ron (Ohms)	1e-3
Machine	
Source frequency (Hz)	60
Inductances (H) [Ld, Lq]	[8.5e-3 8.5e-3]
Resistance (Ohms)	0.2
Flux induced by magnets (Wb)	0.175
Pairs of poles	4

In average model, the drive type will have to be selected depending on the controller. Relevant ones are as follows:

- Induction motor with Field oriented control – select Field oriented control
- PMSM with Vector control – select PMSM vector control
- SVM control for either motor – select space vector modulation

Also the Machine parameters will have to be entered which are same as the ones used for the motor model.

III. Motor control block



Inputs/Outputs:

Torque

The torque reference, typically provided by a speed controller.

teta

The rotor flux angle from the motor output

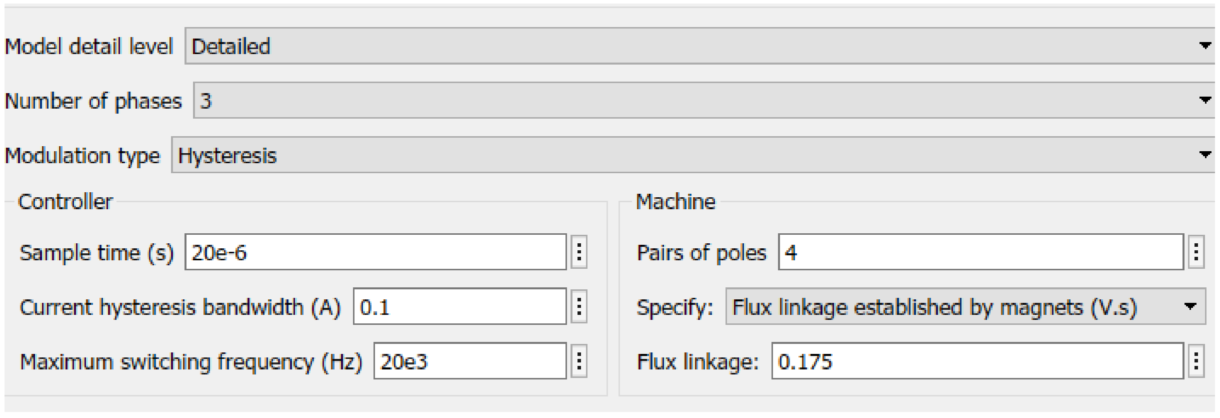
I_abc

The three line currents of the three-phase PMSM. This is measured using current sensor from the inverter output side.

Gates

The pulses for the six inverter switches.

Parameters:



Model detail level: Detailed

Number of phases: 3

Modulation type: Hysteresis

Controller

Sample time (s)	20e-6
Current hysteresis bandwidth (A)	0.1
Maximum switching frequency (Hz)	20e3

Machine

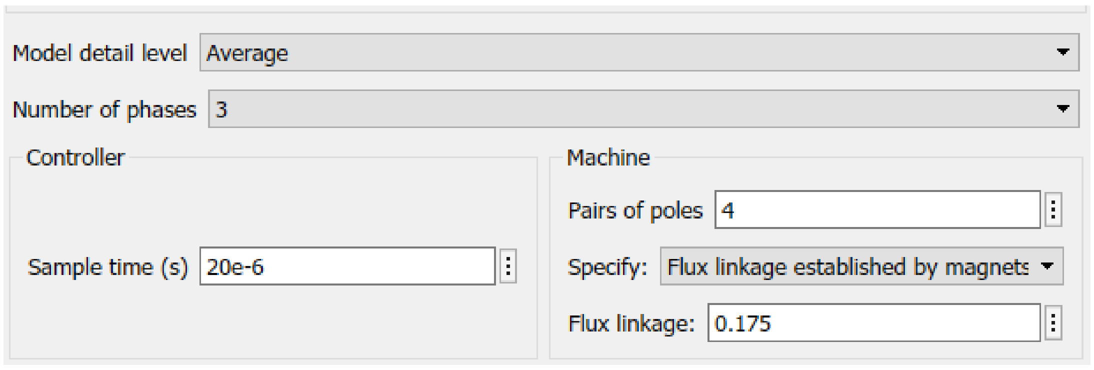
Pairs of poles	4
Specify:	Flux linkage established by magnets (V.s)
Flux linkage:	0.175

In detailed model, the modulation type needs to be selected which can be one of the two as discussed previously:

- Hysteresis (current hysteresis bandwidth needs to be entered in this case)
- SVM (q-d axis PI gains need to be entered)

Machine parameter requires Flux linkage which is same as the motor flux linkage.

In case of average model, only the machine parameters need to be taken care of.


Model detail level: Average

Number of phases: 3

Controller

Sample time (s): 20e-6

Machine

Pairs of poles: 4

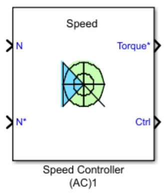
Specify: Flux linkage established by magnets

Flux linkage: 0.175

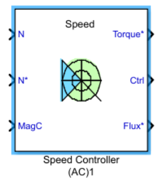
IV. Speed control block

The controller type needs to be selected which can be:

- With flux regulation (use for induction motors with Flux oriented control block)



- Without flux regulation (use for PMSM with vector control block)



<u>Inputs/Outputs:</u>

N

The speed of the machine, in rpm from the motor output.

N*

The speed reference of the machine, in rpm. When the Regulation type parameter is set to **Torque regulation**, this input refers to the torque reference of the machine.

MagC

A binary signal indicating if the machine is magnetized enough to be started (1) or not (0). Provided by FOC block for induction motor. This input is available only if the **Speed controller type** parameter is set to **with flux reference output**.

Torque*

The torque reference, in newton-meter. This is the input to the FOC/ Vector control block.

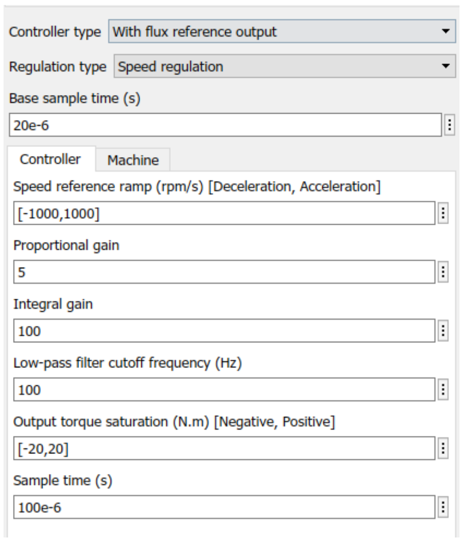
Ctrl

A bus containing measurements.

Flux*

The flux reference, in weber. This output is available only when the **Speed controller type** parameter is set to **with flux reference output**. This is an input to the FOC block in an induction motor drive.

Parameters:


Controller type: With flux reference output

Regulation type: Speed regulation

Base sample time (s): 20e-6

Controller | Machine

Speed reference ramp (rpm/s) [Deceleration, Acceleration]: [-1000,1000]

Proportional gain: 5

Integral gain: 100

Low-pass filter cutoff frequency (Hz): 100

Output torque saturation (N.m) [Negative, Positive]: [-20,20]

Sample time (s): 100e-6

PI gains for the controller are entered. Also the Output Torque Saturation (limit torque to limit current in motor for safety reasons) needs to be entered depending on the motor specification. Note that the sample time can be 5 times the sample time of Control/Inverter block.